DreamCube: RGB-D Panorama Generation via Multi-plane Synchronization Supplementary Material

Yukun Huang[1] Yanning Zhou[2] Jianan Wang[3] Kaiyi Huang[1] Xihui Liu[1†]
[1]The University of Hong Kong [2]Tencent [3]Astribot
https://yukun-huang.github.io/DreamCube/

1. Implementation Details

We implement both Multi-plane Synchronization and DreamCube using PyTorch. For DreamCube, we utilize Stable Diffusion v2 [7] as pre-trained backbone. At training time, we adopt the DDPM noise scheduler [2] with 1000 timesteps. We use a batch size of 4 for training, where the resolution of RGB images and depth maps is 512×512. Random rotation and flipping are used to expand the amount and diversity of panorama training data. The depth rescaling parameter s is randomly sampled from a uniform distribution in $[0.2, 1.0]$ during training. We froze the VAE and fine-tuned the diffusion U-Net for 10 epochs. We use the AdamW optimizer with a learning rate of $2 \cdot 10^{-5}$. The entire training process took approximately two days on four Nvidia L40S GPUs. At inference time, we adopt the DDIM noise scheduler [8] with 50 sampling steps. The depth rescaling parameter s is fixed to 0.6 for inference.

2. Data Processing Pipeline

All panorama data needs to be processed into a unified format, including RGB cubemaps, depth cubemaps, and image captions for each cube face. For datasets not originally in cubemap format, we apply standard perspective projection to produce cubemaps. Next, we adopt BLIP-2 [5] to obtain image captions of all cube faces. While the Structured3D dataset includes depth data, the other datasets only contain RGB data. To annotate the depth of these panoramas, we build a high-resolution panorama depth estimation pipeline by connecting the existing panorama depth estimation work Depth Anywhere [9] and the image-guided depth upsampling work PromptDA [6], which supports panoramic depth estimation at 4K resolution. We use this pipeline to perform depth estimation on equirectangular-based panoramas and then project the obtained depth panoramas into cubemaps.

† Corresponding author.

3. More Analysis

We further provide efficiency, ablation, robustness, and generalization analyses to evaluate the proposed method.

Efficiency analysis. We provide an efficiency analysis of our approach in Table 1 compared to the baseline model, Stable Diffusion v2 (SD2) [7]. Among all synchronized operators, synchronized Self-Attention ("+SyncSA") incurs the most computational cost, increasing TFLOPs by 76.1% and latency (ms) by 113.1% than no synchronization ("No Sync."). This accounts for 86.0% of the latency cost and almost 100% of the TFLOPs cost incurred by our approach.

Table 1. **Efficiency analysis.** We evaluate the computational efficiency of SD2's and DreamCube's U-Nets in a single forward pass and report the metrics: TFLOPs and Latency (ms).

Methods		FLOPs (T)	Latency (ms)
SD2's U-Net	batch-size=1	0.804	35.4
	batch-size=6	4.826	138.8
DreamCube's U-Net	No Sync.	4.827	139.4
	+SyncSA	8.502	297.1
	+SyncConv	4.827	144.3
	+SyncGN	4.827	152.0
	All Sync.	8.502	322.7

Ablation analysis of DreamCube. We analyze different components of DreamCube, with results shown in Table 2. We evaluate both RGB and depth panorama generation on the Structured3D test split [12], following the standard evaluation protocol. Both XYZ Positional Encoding ("XYZ Pos.") and Multi-plane Synchronization ("Sync.") improve performance, with "Sync." yielding the most substantial gains, which reduces FID by 8.77 and improves δ-1.25 by 0.103. Specifically, Synced Self-Attention ("SyncSA") contributes the most performance gain compared to other synced operators. Besides, we further provide a qualitative ablation analysis of XYZ Positional Encoding in Figure 1. Our design effectively alleviates line artifacts and content

Table 2. **Ablation analysis of DreamCube**, where the performance evaluation is performed on the Structured3D test split [12]. Both proposed Multi-plane Synchronization and XYZ Positional encoding bring performance improvements.

Methods	RGB		Depth	
	FID ↓	IS ↑	δ-1.25 ↑	AbsRel ↓
w/o XYZ Pos.	13.66	5.57	0.784	0.136
w/o Sync.	21.35	5.62	0.684	0.168
w/o SyncSA	24.38	**5.78**	0.715	0.158
w/o SyncConv	19.62	5.60	0.779	0.139
w/o SyncGN	18.35	5.51	0.784	0.135
DreamCube (Ours)	**12.58**	5.50	**0.787**	**0.133**

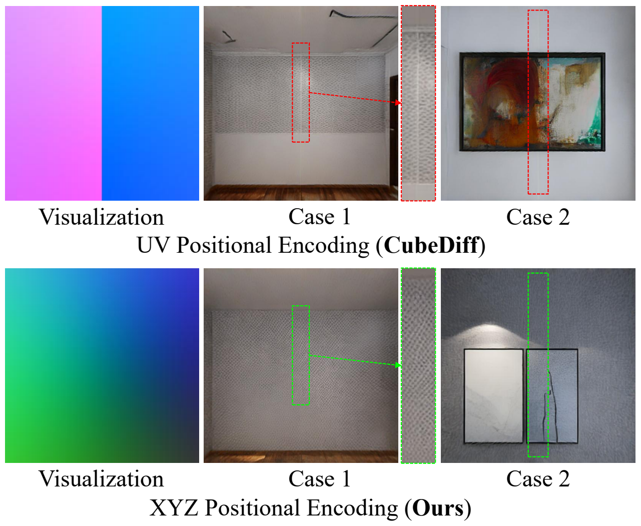
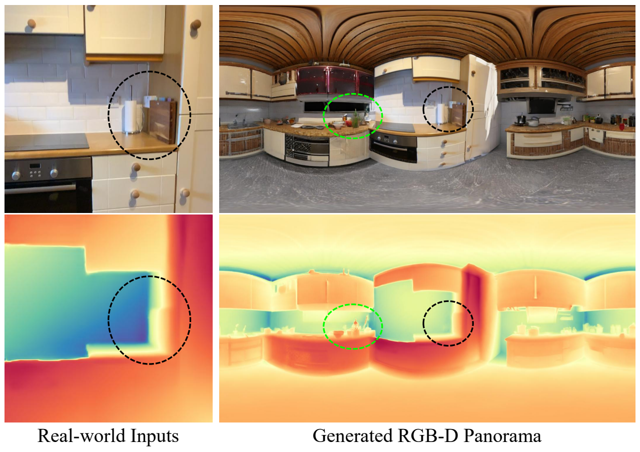
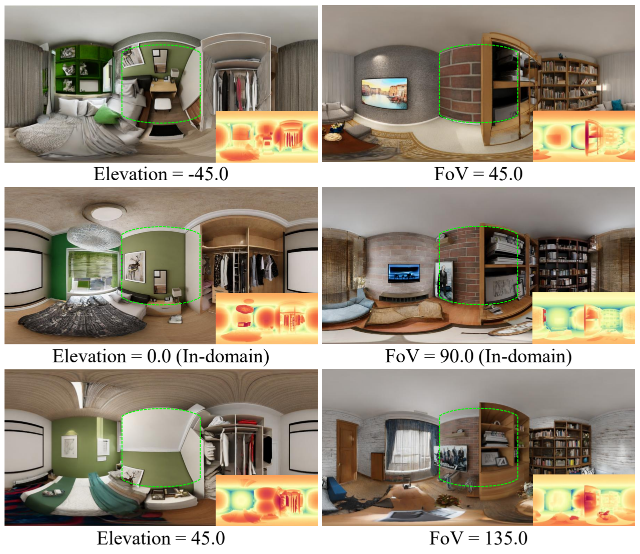

Visualization
Case 1
Case 2
UV Positional Encoding (**CubeDiff**)
Visualization
Case 1
Case 2
XYZ Positional Encoding (**Ours**)

Figure 1. **Ablation analysis of XYZ Positional Encoding.** We present the qualitative results of the back view of cubemap, where the UV positional encoding introduces discontinuous numerical steps. This leads to line artifacts (Case 1) and incoherent visual contents (Case 2), as indicated by the **red** dashed box. In contrast, our proposed XYZ positional encoding alleviates these issues in both cases, as shown within the **green** dashed box.



(a) Generated results from real-world inputs captured by sensors [1]. Even though the input depth is low-resolution (as indicated by the **black** dashed circles), our method is still able to generate high-definition depth maps (as indicated by the **green** dashed circles).



(b) Generated results from inputs with extreme viewing angles, where the **green** dashed boxes highlight the input views.

Figure 2. **Robustness analysis** of DreamCube to out-domain RGB-D inputs from real world and extreme viewing angles.

incoherence compared to UV Positional Encoding.

Robustness analysis of DreamCube. DreamCube takes single-view RGB-D images as input for cubemap generation. To evaluate the robustness of DreamCube, we test various types of RGB-D inputs and provide the generated results in Figure 2. Specifically, we test real-world inputs captured by sensors [1]. Unlike synthetic training data, real-world inputs have low-resolution depth maps and non-standard camera views. Even so, our method is still able to generate reasonable panoramas with high-resolution depth maps, as shown in Figure 2a. In addition, we also test inputs with extreme camera views (*e.g.*, elevation and FoV). DreamCube struggles to generate correct panoramas under inputs with extreme elevation angles, but shows robustness to perturbations of the FoV, as shown in Figure 2b.

Generalization analysis of DreamCube. To evaluate DreamCube's generalization capabilities, we present out-domain generation results in Figure 3, where the input RGB images are generated from Flux.1-dev [4]. We obtain the corresponding input depth map using Depth Anything v2 [10]. Despite the significant domain gap between these inputs and our training distribution, DreamCube successfully generates coherent and visually plausible RGB-D panoramas, demonstrating its strong generalization ability.

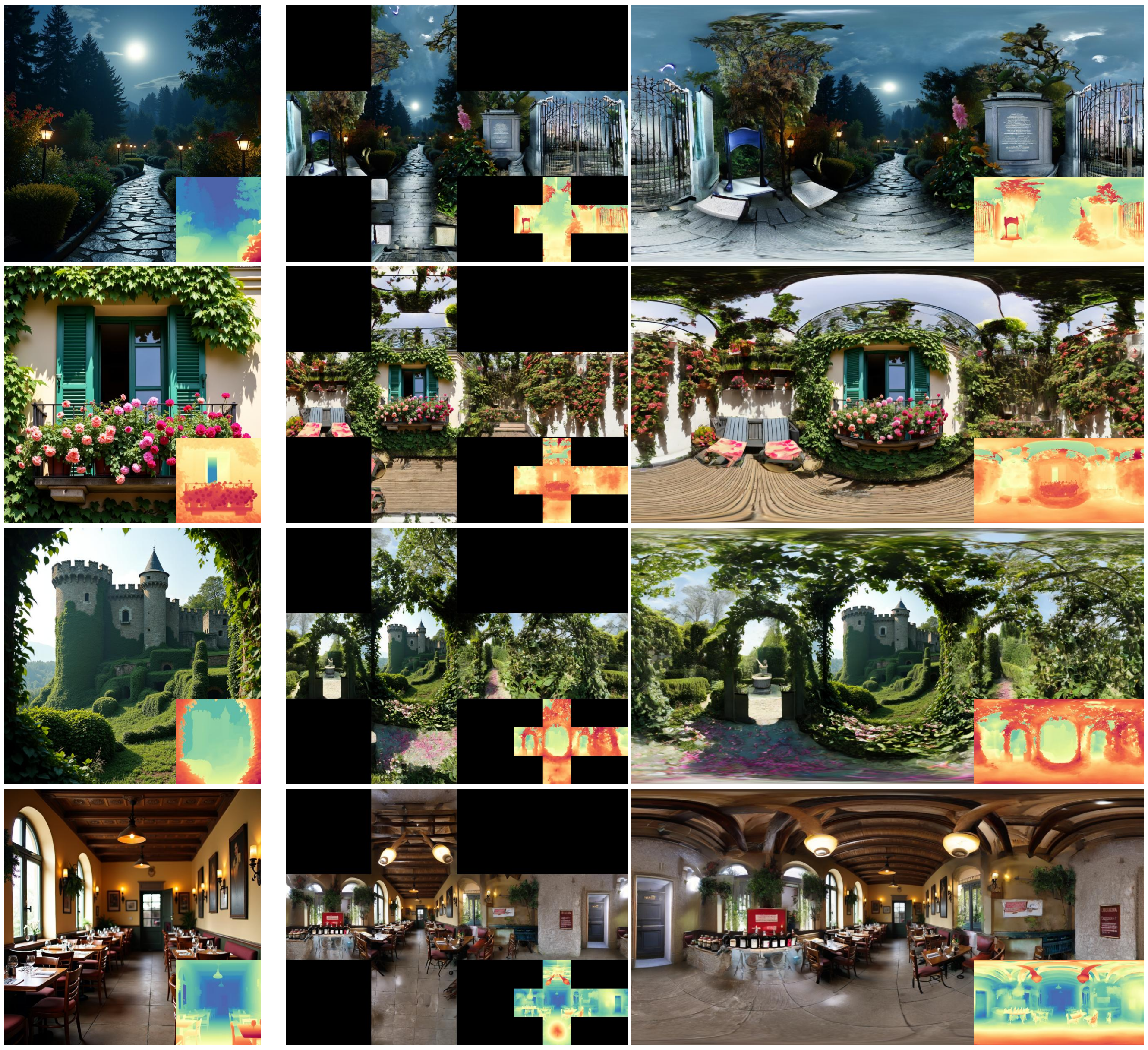


Figure 3. **Out-domain RGB-D panorama generation.** The RGB-D inputs are obtained by Flux.1-dev [4] and Depth Anything v2 [10]. DreamCube demonstrates generalization ability on diverse inputs, maintaining high-quality RGB appearance and geometric consistency.

4. Panorama-to-3D Reconstruction

4.1. Implementation

The generated RGB-D panoramas contain the direction and distance of each pixel, so a colored 3D point cloud can be obtained by projecting all pixels into 3D space. We can further convert the point cloud into different 3D representations, such as 3D meshes and 3D Gaussians [3]. Note that these conversions can be achieved either by differentiable optimization or by handcrafted algorithms. We choose handcrafted algorithms for fast 3D scene reconstruction in seconds from RGB-D panoramas. Specifically, for 3D mesh reconstruction, we use the obtained point cloud as vertices, and the vertex colors are assigned by the RGB values of the corresponding pixels. The connections between vertices can be extracted based on the adjacency relationship of image pixels. For 3D Gaussians, the position and color of each Gaussian point can be directly assigned from the colored point cloud, while other properties are inferred using a method similar to WonderWorld [11].

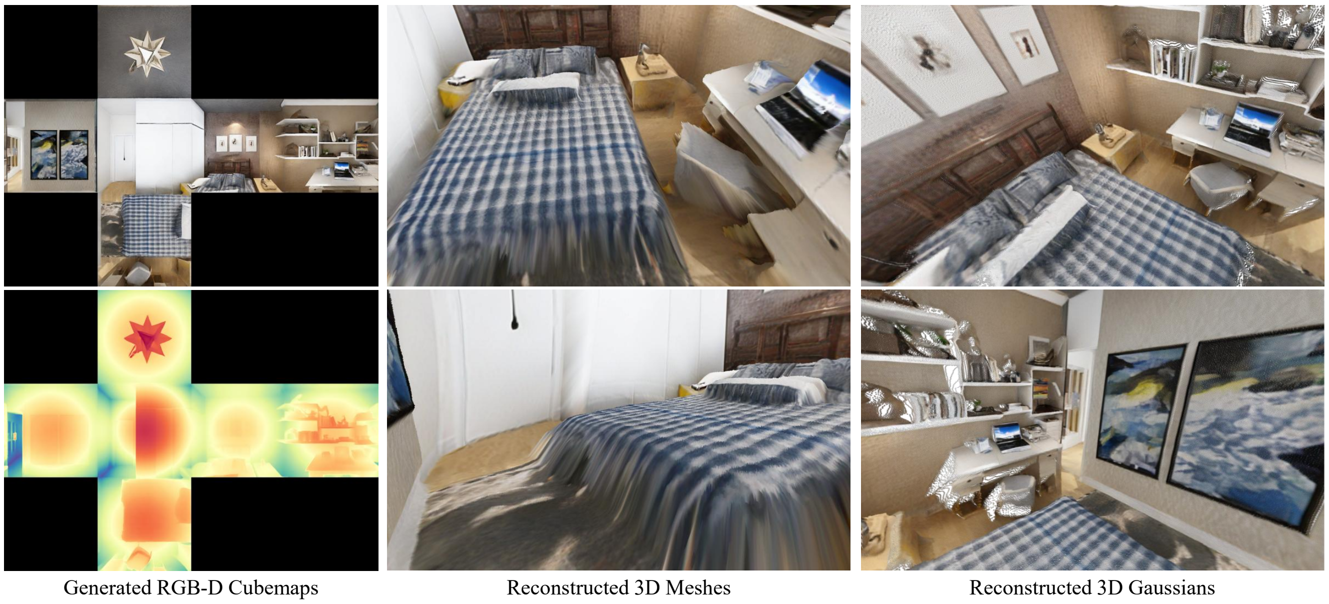


Figure 4. **Panorama-to-3D scene reconstruction.** Based on the RGB-D cubemap generated by DreamCube, we can reconstruct the corresponding 3D scenes in seconds and obtain both 3D mesh and 3D Gaussian [3] representations.

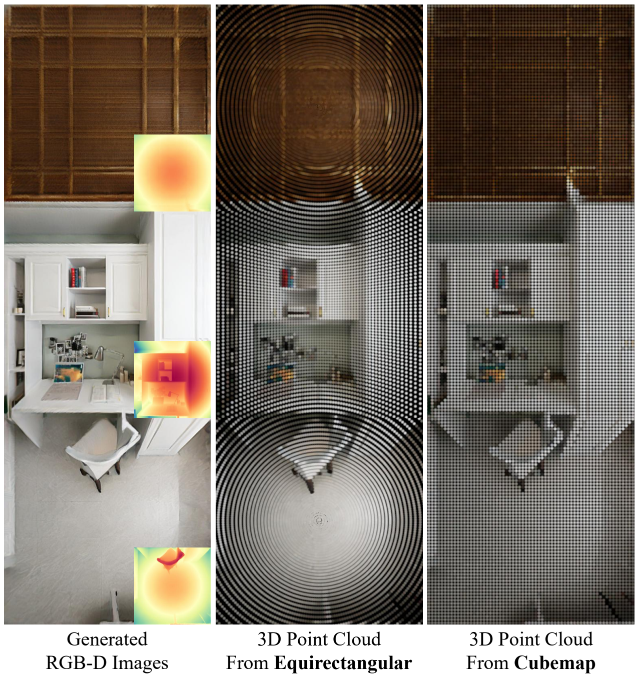

Generated RGB-D Images
3D Point Cloud From **Equirectangular**
3D Point Cloud From **Cubemap**

Figure 5. **Qualitative comparison of 3D point clouds** reconstructed from equirectangular-based and cubemap-based RGB-D panoramas. Equirectangular panoramas produce an uneven, ring-shaped 3D point distribution dense near the poles, while cubemap panoramas yield a more uniform and regular distribution.

It is worth mentioning that different panoramic projections affect the quality of the reconstructed 3D scene. For equirectangular-based RGB-D panoramas, due to the significant geometric distortion at the poles, the reconstructed 3D point cloud will be unevenly distributed and particularly dense at the top and bottom poles. In contrast, the distribution of 3D points from RGB-D cubemap is more uniform and regular, as shown in Figure 5.

4.2. Results

An important application of DreamCube is fast 3D scene generation. Benefiting from the joint RGB-D panorama generation model and the rapid panorama-to-3D projection algorithm, our approach can achieve 3D scene generation from a single view in about ten seconds. We present the visualized results of the generated 3D scenes in both 3D meshes and 3D Gaussian representations, as shown in Figure 4. The visual quality of the reconstructed 3D scene is comparable to that of the original panorama. Additionally, we analyze the impact of different formats of RGB-D panoramas on 3D reconstruction, as illustrated in Figure 5. The 3D point distribution derived from equirectangular-based panoramas is uneven, exhibiting a ring-shaped pattern with particularly dense points near the poles. In contrast, the 3D point distribution from cubemap-based panoramas tends to be more uniform and regular.

5. Limitations

Limitations of our method include high computational cost and restricted input conditions. First, DreamCube samples six image latents simultaneously, which increases computational demands and limits the scalability of training batches.

However, compared to existing panorama generation methods, our method has the superior computational utilization (effective pixel ratio obtained at the same computational cost), because it uses a less distorted cubemap instead of equirectangular, and does not require redundant FoV overlapping for seam continuity. Second, DreamCube is trained with cubemap's front face as input conditions. When the input distribution deviates from the training domain, for example, non-frontal view or extreme FoV, the generated results may fail, as shown in Figure 2.

References

[1] Gilad Baruch, Zhuoyuan Chen, Afshin Dehghan, Tal Dimry, Yuri Feigin, Peter Fu, Thomas Gebauer, Brandon Joffe, Daniel Kurz, Arik Schwartz, and Elad Shulman. ARKitScenes - A Diverse Real-World Dataset for 3D Indoor Scene Understanding Using Mobile RGB-D Data. In *Neural Information Processing Systems*, 2021. 2

[2] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising Diffusion Probabilistic Models. *Advances in Neural Information Processing Systems*, 33:6840–6851, 2020. 1

[3] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3D Gaussian Splatting for Real-Time Radiance Field Rendering. *ACM Transactions on Graphics*, 42(4), 2023. 3, 4

[4] Black Forest Labs. Flux.1-dev. `https://huggingface.co/black-forest-labs/FLUX.1-dev`, 2025. Accessed: 2025-01-19. 2, 3

[5] Junnan Li, Dongxu Li, Silvio Savarese, and Steven Hoi. Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models. In *International conference on machine learning*, pages 19730–19742. PMLR, 2023. 1

[6] Haotong Lin, Sida Peng, Jingxiao Chen, Songyou Peng, Jiaming Sun, Minghuan Liu, Hujun Bao, Jiashi Feng, Xiaowei Zhou, and Bingyi Kang. Prompting depth anything for 4k resolution accurate metric depth estimation. *arXiv preprint arXiv:2412.14015*, 2024. 1

[7] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-Resolution Image Synthesis with Latent Diffusion Models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10684–10695, 2022. 1

[8] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising Diffusion Implicit Models. In *International Conference on Learning Representations*, 2021. 1

[9] Ning-Hsu Albert Wang and Yu-Lun Liu. Depth Anywhere: Enhancing 360 Monocular Depth Estimation via Perspective Distillation and Unlabeled Data Augmentation. *Advances in Neural Information Processing Systems*, 37: 127739–127764, 2024. 1

[10] Lihe Yang, Bingyi Kang, Zilong Huang, Zhen Zhao, Xiaogang Xu, Jiashi Feng, and Hengshuang Zhao. Depth Anything V2. *Advances in Neural Information Processing Systems*, 37:21875–21911, 2024. 2, 3

[11] Hong-Xing Yu, Haoyi Duan, Charles Herrmann, William T Freeman, and Jiajun Wu. Wonderworld: Interactive 3d scene generation from a single image. *arXiv preprint arXiv:2406.09394*, 2024. 3

[12] Jia Zheng, Junfei Zhang, Jing Li, Rui Tang, Shenghua Gao, and Zihan Zhou. Structured3D: A Large Photo-Realistic Dataset for Structured 3D Modeling. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part IX 16*, pages 519–535. Springer, 2020. 1, 2